

Design of a microprogrammable computer with bit-slice devices

Dogan Ibrahim outlines the basic design principles for a 16-bit microprogrammable computer using bit-slice devices

In this two-part series of articles the basic principles of designing a 16-bit computer made up of 4-bit bit-slice devices are discussed. The first part considers the data processing and control sections of such a system.

Fixed instruction set microprocessors have been very popular for the last five years and have been increasingly used in new applications. These microprocessors are manufactured using MOS technology and all data processing and control logic are implemented on a single chip. In some cases, the program memory, data memory and input/output channels are included in the same chip, to form a single chip microcomputer. Because of their low cost and small size, the fixed instruction set microprocessors have replaced random logic with programmable logic, leading to an increase in overall system reliability and flexibility. The user can simply change the software (usually stored in ROM or EPROM) for a new application.

The major disadvantage of the MOS microprocessors is their relatively low speed which prevents their application to areas such as signal processing, high speed disc controllers etc.

Bit-slice microprocessors on the other hand are manufactured using bipolar Schottky technology and differ from MOS processors in the architecture of their CPUs. The data processing and the control logic are realized on separate chips giving the final system much more flexibility and an increase in the execution rates by almost a factor of ten. The user of a bit-slice computer can invent his own instruction set and implement it in the computer's control memory (microprogram memory). The microprogram memory is N words deep by M bits wide (N and M selected by the computer designer) and for every machine instruction (macroinstruction) there exists a unique sequence of instructions (microinstructions) residing in the microprogram memory. The data output from the microprogram memory is distributed to most parts of the computer and these constitute the control signals. The instruction set of the computer can be changed by reprogramming the microprogram memory. The width of a microprogram

memory depends upon the complexity of the hardware but it generally ranges from 30 to 70 bits. The number of words in a microprogram memory depends upon the number of microinstructions it is wished to add to the computer and ranges typically from 256 to 2k words.

The main advantages of the microprogrammable computers over the fixed instruction (MOS) computers are:

- higher speed of operation because of their bipolar technology
- special instructions can be implemented for specific applications, which not only increases the rate of operation but also simplifies assembly level programming
- the instruction sets of other computers can be emulated by appropriate programming of the microprogram memory
- the designer of a bit-slice computer can choose and modify the word length according to the specific application by cascading several bit-slice devices (e.g. word lengths of 4, 8, 12, 16, . . . bits are all possible)
- CPU hardware can easily be modified, making bit-slice computers much more flexible and adaptable
- the user of a bit-slice computer gains a better understanding of the internal architecture of computers

Because of the chip complexity, high power dissipation and speed, the bit-slice devices are usually manufactured in sections of 4-bit wordlengths and special pins are brought out of the chip for cascading to any required word length.

The following section of this article will describe the design of a 16-bit computer using bit-slice devices.

MICROPROGRAMMABLE COMPUTER ARCHITECTURE

The general architecture of a microprogrammable computer is shown in Figure 1. The arithmetic processing unit (APU), computer control unit (CCU), interrupt control unit (ICU), program counter unit (PCU) and the timing unit (TU) are all subsystems of the central processing unit (CPU).

Department of Electrical & Electronic Engineering, The City University, Northampton Square, London, EC1V 0HB, UK

Data and instructions are transmitted along the 16-bit bidirectional data bus. Addresses for the memory or for an input/output device are sent along the 16-bit address bus.

The computer control unit is the part of the CPU that controls all the other subsystems; fetches and decodes the instructions residing in the memory; and sends the appropriate signals to other parts of the computer to perform the required operation.

The arithmetic processing unit is the subsystem where computing and data manipulation are carried out. Arithmetic and logical operations, data shifting and rotation and the generation of status signals are all functions of the APU.

The program control unit generates the memory addresses; steps through the addresses in the memory; and performs branch, conditional branch and subroutine call and return instructions under the control of the CCU.

DESIGN OF THE APU

The APU consists of three subsystems: the arithmetic and logic unit (ALU), the data shifting logic (DSL) and the condition code select logic (CCS). The ALU is the unit where all of the arithmetic and logical operations are performed. The DSL is the data shifting and rotating logic external to the ALU, while the CCS stores the ALU status bits, selects the required status under the control of the CCU and sends them to the CCU for processing conditional operations (conditional branches — branch on zero etc.)

Arithmetic logic unit

The requirements of an ALU in a microprogrammed

design are that:

- it should be capable of executing the required logical and arithmetic operations
- it should be programmable so that any required operation can be executed under the control of the CCU
- cascading to any number of bits should be feasible
- the speed should be such that the operations can be executed at the maximum desired clock frequency

There are several logic unit/function generators that satisfy some or all of the above requirements, but the one chosen here is the Am2903 bipolar bit-slice ALU, manufactured by the Advanced Micro Devices.

The Am2903 is a 4-bit cascable bipolar logic unit/function generator that can perform operations on two 4-bit input variables. It contains 16 internal working registers and in addition to its arithmetic and logic operations, the Am2903 provides special instructions for multiplication, division, normalization etc.

The block diagram of the Am2903 is shown in Figure 2. The source, function and the destination of any external or internal data are controlled by nine bits, I_0-I_8 . The device consists of a 16-word by 4-bit, two port RAM which serves as the internal working registers. The RAM

Table 1. Programming for slice position

Am2903 position	LSS	WRITE/MSS
Least significant device	low	—
Intermediate devices	high	high
Most significant device	high	low

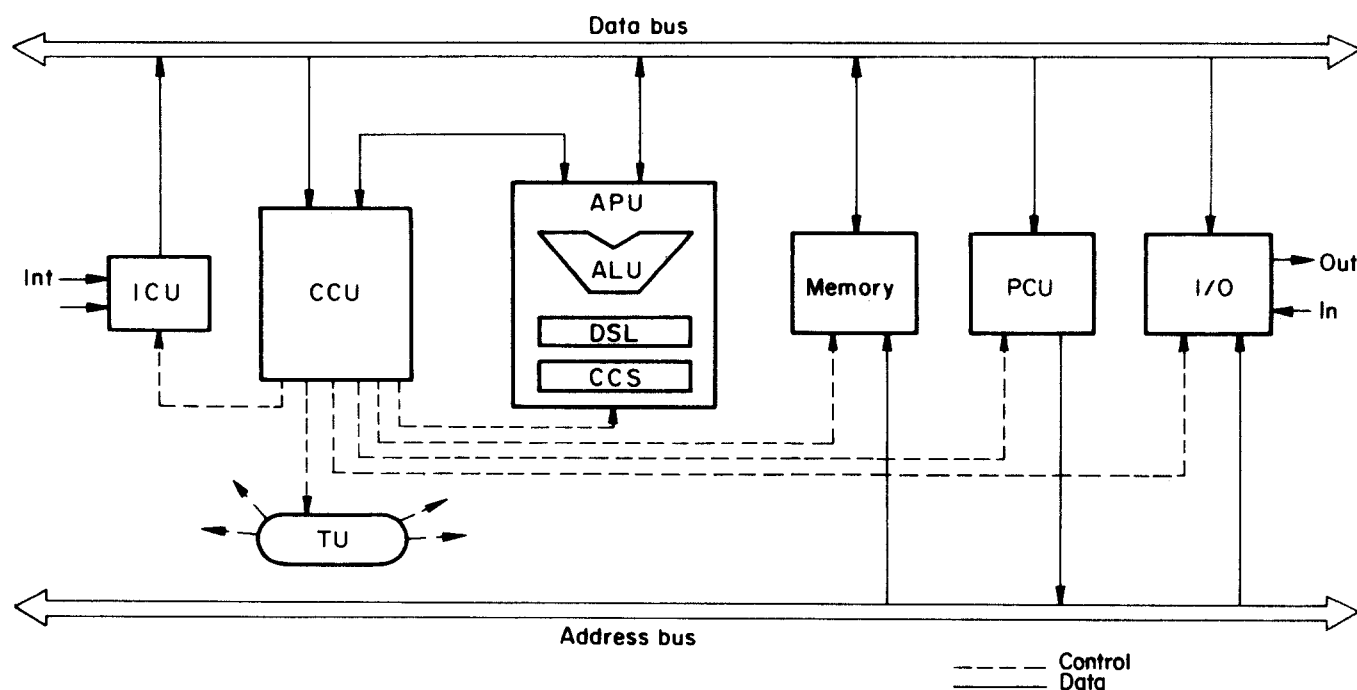


Figure 1. Architecture of a microprogrammable computer

words are addressed by the A and B address ports (four bits each). Data is written into any RAM location selected by the B address port, when the write enable input, $\overline{WE}$, the write output, $\overline{WRITE}$, and the clock C_p , are all low. In addition to the 16 words of RAM, the Am2903 also contains a Q register which is 4-bits wide and it can be used as a general purpose register. The DB and Y ports are 4-bit bidirectional ports and are driven by tristate buffers. When $\overline{OE_Y}$ is low, the Y outputs are enabled and are in a high impedance state when $\overline{OE_Y}$ is high. When $\overline{OE_B}$ is low, the DB outputs are enabled and are in a high impedance state when $\overline{OE_B}$ is high.

The Am2903 also generates four status signals: C_{n+4} is the carry out pin and in the active state it indicates that the operation exceeded the 4-bit number range. The OVR and N outputs indicate the overflow and the sign of the operation. The Z output is open collector and it can be wire- or'd between cascaded devices to indicate the zero status.

For a 16-bit computer, the APU should consist of four Am2903s cascaded in the correct manner. The slice position is programmed via the LSS and the WRITE/MSS pins as shown in Table 1.

Am2903s can be cascaded by two methods, either by ripple carry connection or by using a carry look-ahead generator device. In the former, the carry out of one device is connected to the carry in of the next higher positioned device. This scheme is not advisable because the speed of operation is limited since the carry has to propagate through all of the 16 locations before the final result settles down. The carry look-ahead generator should be used where high speed is an important criterion. Here, the $\overline{G}$ and $\overline{P}$ outputs of each Am2903 are connected to the appropriate $\overline{G}$ and $\overline{P}$ inputs of the carry look-ahead, and the C_{n+x} , C_{n+y} , C_{n+z} outputs of the carry look-ahead generator are connected to the appropriate carry in (C_n) inputs of Am2903s.

The logic diagram of a 16-bit ALU together with the carry look-ahead device, Am2902, is shown in Figure 3. In this diagram the $\overline{WRITE}/MSS$ output of the least significant device is connected to the $\overline{WE}$ pins of all the other devices. Thus, writing into RAM is controlled by the $\overline{IEN}$ pin. The truth table (Table 2) shows how the RAM is controlled via the $\overline{IEN}$ pin.

Data shifting logic

This is the logic external to ALU that shifts data to either end of the ALU registers, under the control of CCU and the ALU control bits $I_5 - I_8$. Shifting can either be in logical or in arithmetic mode. In logical shifting, the sign bit (most significant bit of the most significant slice) is also shifted left or right, whereas in the arithmetic shifting mode, the sign bit is preserved and the other bits are shifted. The Q register is always shifted in logical mode. There is also the possibility of shifting the RAM and the Q register simultaneously or shifting either of them separately.

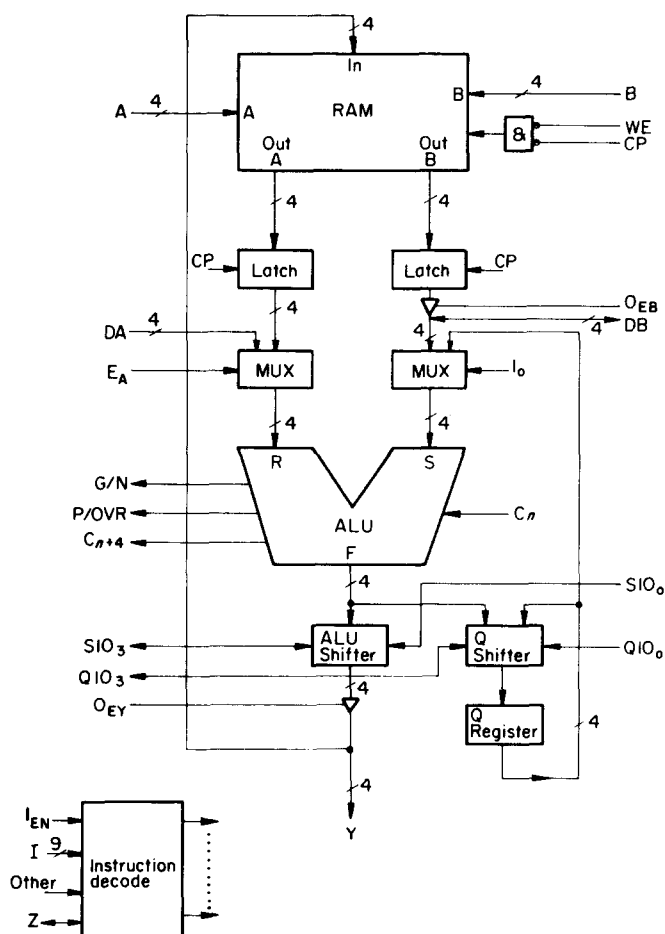


Figure 2. Block diagram of Am2903

Table 2. Truth table for RAM control

$\overline{IEN}$	C_p	$\overline{WE}$	RAM control
L	L	L	Write into RAM
L	H	L	Inhibit writing
H	L	H	Inhibit writing
H	H	H	Inhibit writing

L = low
H = high

The data shifting logic designed here uses two low power Schottky dual 4-line to 1-line data multiplexers (Am25LS253) and an eight-to-one multiplexer (Am25LS151). The logic of the DSL is shown in Figure 4. Under the control of the CCU one of eight data (D_0 to D_7) can be shifted into any ends of the ALU registers. DSL is controlled by the CCU with the five control signals S_1 to S_5 . With only four LSI and a few MSI logic devices the DSL shown here is capable of performing over 100 different shift operations

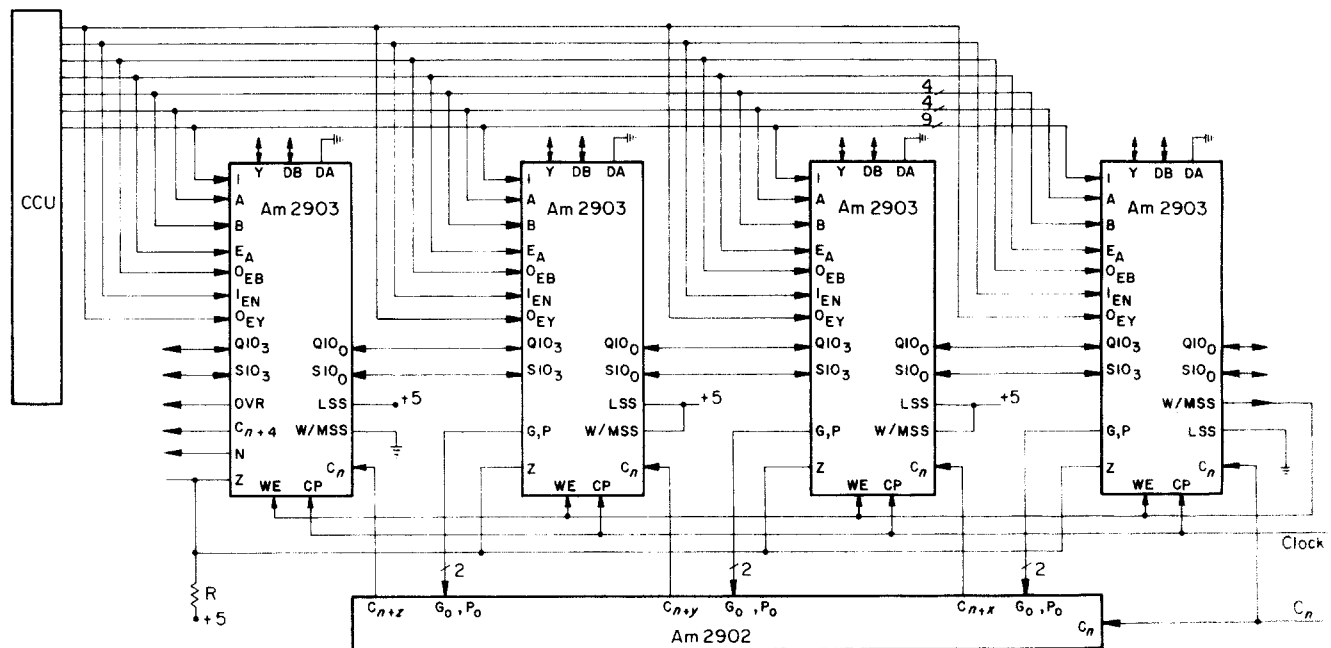


Figure 3. Connection of Am2903s for a 16-bit ALU

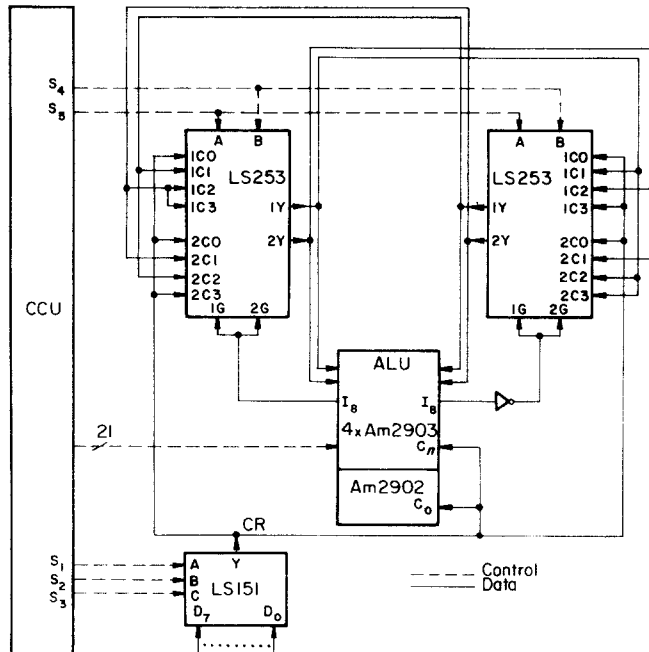


Figure 4. Data shifting logic around Am2903s

(compare with the fixed instruction microprocessor data shifting facilities). Notice in Figure 4 that the carry in (C_n) to the ALU is connected to the CR output of the shift multiplexer. Thus, double length and conditional arithmetic can be performed easily by choosing one of the eight data as the source for the carry in input. The same

control signals (S_1, S_2, S_3) control the carry in and the CR.

Condition code select logic

This is the logic that manipulates the status bits of the ALU and sends them to the computer control unit to perform conditional or unconditional operations. The logic diagram of the CCS is shown in Figure 5. The source of data to the status register is derived either from the four ALU status bits or from the highest four bits of the data bus. Thus the status register can either be loaded with the normal status bits or it can be loaded with external data (from the memory or ALU registers). The output of the status register with the correct polarity is sent to the CCU for performing the conditional operations. Provision is also made for the status register data to be saved in the memory or in the ALU registers via the data bus.

The 16 DA input lines of the Am2903 can be connected in several ways. They can either be controlled from a PROM, in which case special bit manipulation functions (bit set, bit reset etc.) can be performed or they can be connected to a constant logic level. Figure 3 shows the case where all 16 DA bits are connected to ground.

COMPUTER CONTROL UNIT

The CCU is the most complicated subsystem in any computer. It is this system that controls the other subsystems, synchronizes the internal and external events and fetches and decodes the instructions residing in the main memory. The tasks of the CCU are described by the following flow table:

initialize the processor
 loop 1 if interrupt detected do interrupt routine
 loop 2 fetch an instruction
 decode the instruction
 increment the program counter
 if another operand required go to loop 2
 execute the instruction
 go to loop 1
 interrupt routine
 disable further interrupts
 service the current interrupt
 go to loop 1

The block diagram of the computer control unit is presented in Figure 6. The instruction register (a 16-bit clocked register) receives the instructions from the memory via the data bus. The instructions consist of two 8-bit fields: the 'opcode' and the 'operand'. The operand is usually the number of a selected register in the ALU and the opcode specifies the operation to be performed.

There is never one-to-one correspondence between the opcodes and the microprogram starting addresses that execute the opcodes. This is because each macroinstruction (machine code) is derived by combining more than one microinstruction in the microprogram memory. Also different types of macroinstructions usually require a different number of microinstructions. To avoid this problem a memory decoder, mapping ROM, is used which decodes the opcodes to their starting addresses in the microprogram memory. The width of the mapping ROM should be equal to or wider than the width of the opcode register. A usual choice is to make the mapping ROM as 12 bits for an 8-bit opcode. This allows the opcodes to be decoded up to 4k in the microprogram memory.

The output of the mapping ROM is loaded into the microprogram sequencer which points to the first microinstruction of the current macroinstruction. The main purpose of the sequencer is to present an address to the

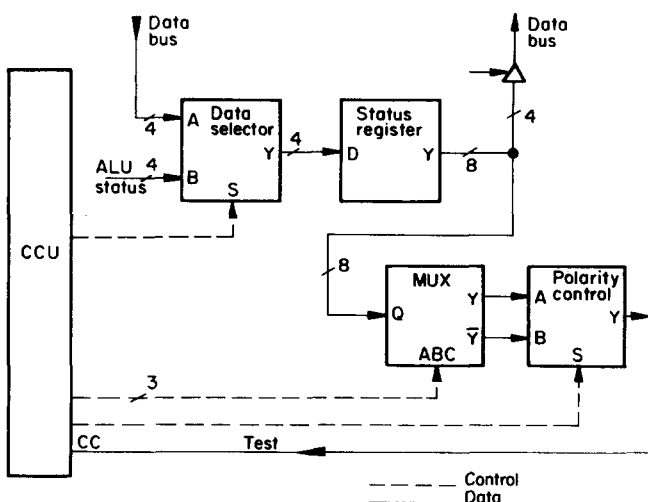


Figure 5. Condition code select logic

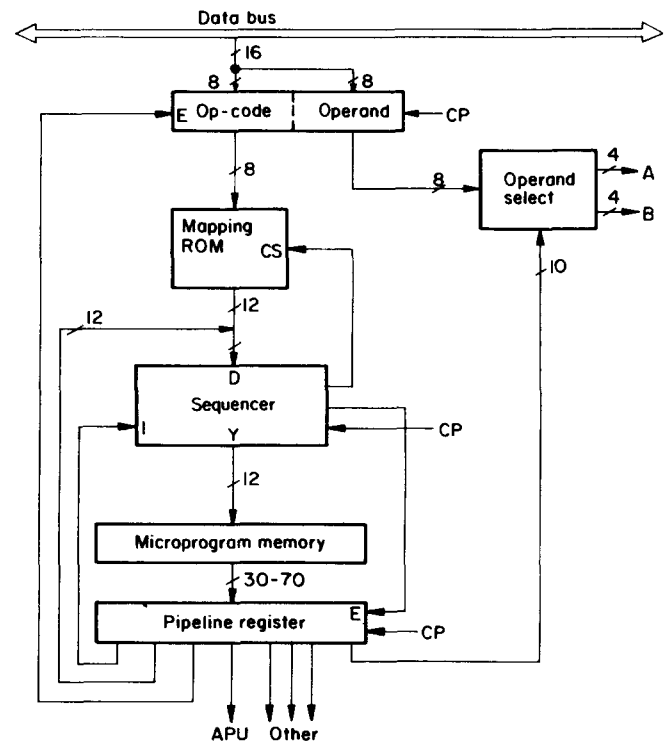


Figure 6. Computer control unit

microprogram memory so that a macroinstruction can be fetched and executed. There are several LSI microprogram sequencers available manufactured by different companies¹.

The Am2910 is a bipolar microprogrammable sequencer that can execute 16 different control instructions (most of them conditional). It can be used to address up to 4096 words of microprogram memory. Basically it contains a 4-input multiplexer which selects the source of the next microinstruction address. In addition to its sequencing and branching functions, the Am2910 provides microsubroutine and looping capability. A 5-word-deep stack saves the microsubroutine return addresses.

The D input is used for branching and the branch address will be available at the tristate Y outputs. The CC input is used during the conditional instructions and the test is applied to this pin. Figure 7 shows the block diagram of the Am2910. The best way to understand the Am2910 is to look at a few examples (see Figure 8). Instruction 0 is the 'Jump Zero' instruction and the output of the sequencer is forced to zero from the current value. This instruction is used during initialization of the processor where address zero in the microprogram memory will contain the instruction 'fetch microcode'. Instruction 14 (continue) is used for single stepping through the microprogram addresses. Instruction 2 is an unconditional instruction and upon execution, the mapping ROM will be enabled and the output of the sequencer will jump to the decoded address. This instruction is used during the fetch cycle. Instruction 3 is a conditional jump instruction and the branch address should be applied to the D inputs. During the execution of this instruction the mapping ROM is dis-

abled to avoid two levels of data at the D inputs. If the condition tested is true ($\overline{CC}$ input low) then the value currently at the D inputs will be selected at the Y outputs of the

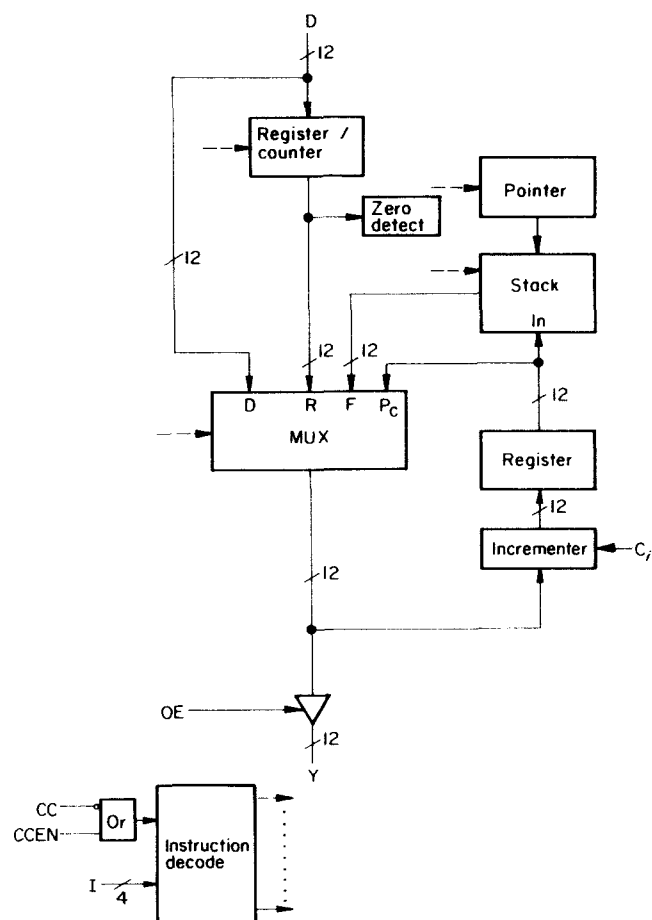


Figure 7. Block diagram of Am2910 microprogram sequencer

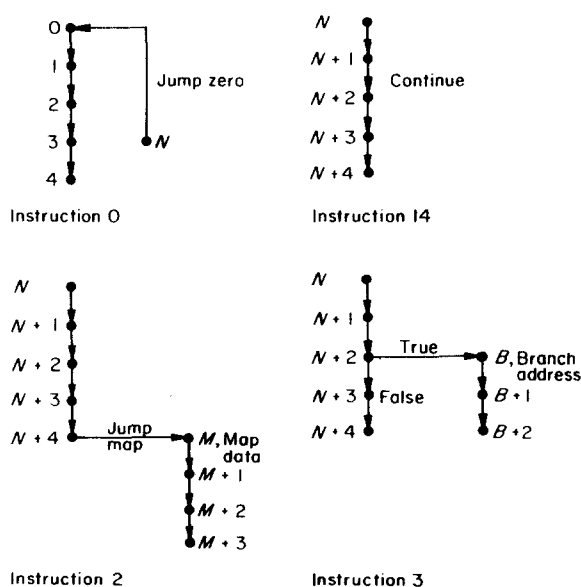


Figure 8. Some Am2910 sequencer instructions (arrows indicate the next sequencer output data)

sequencer otherwise the output of the sequencer will be incremented by one.

The output of the sequencer is the address input to the microprogram memory, which is a fast bipolar ROM/PROM. It is not possible at this stage to determine the required width of the microprogram memory and it can be fixed when the remaining subsystems of the computer are designed.

Microprogram data is loaded into a clocked register (pipeline register), which speeds up the operation by allowing the address of the microprogram memory to be changed during execution of the current microinstruction (at the output of the pipeline). It is important to understand that there is no sequential logic between the sequencer and the pipeline register input. Thus, when the sequencer control bits are programmed by the current microinstruction, the next required microinstruction data will be available at the output of the microprogram memory and it will be distributed to the computer hardware on the next clock cycle.

The operand select logic provides the register addresses (A and B) to the Am2903s. Some macroinstructions may only require a single 4-bit operand (such as incrementing a register) and some may require two 4-bit operands, such as $R_1 + R_2 \rightarrow R_2$, where the operand field may take the hexadecimal value 12H. Provision can also be made to select any register entirely by the CCU without having to specify any operand in the macroinstruction. The operand select logic is controlled by two bits and the source of data to the A and B address fields are either derived from the microprogram memory or from the operand field of the instruction register.

CONCLUSION

This article has described the basic design principles of the data processing and control section of a 16-bit microprogrammable computer. The design principles outlined here are not unique and there are several factors that the design engineer has to consider before the overall system is wired up. Next month's article will describe the design principles of the remaining subsystems of the computer.

ACKNOWLEDGEMENT

The author wishes to thank the UK Science Research Council for financial support.

REFERENCES

- 1 Abxandius, N A 'Bit-sliced microprocessor architecture' *Computer* Vol 11 No 6 (June 1978)
- 2 Husson, S S *Microprogramming principles and practices* Prentice-Hall, USA (1970)
- 3 Katzan, H *Microprogramming primer* McGraw-Hill, New York (1977)

BIBLIOGRAPHY

The Am2900 family data book Advanced Micro Devices inc., Sunnyvale, California, USA (1978)